Assignment 4

Extended Kalman filter

- **Extended Kalman filter (EKF)** is the nonlinear version of the Kalman filter which linearizes about an estimate of the current mean and covariance. For more details, see this.

- In this assignment, you'll use the same bag as assignment 3 (***sdc_hw3.bag***). The bag contains two type of message:

  imudata (sensor_msgs/Imu)
  visual odometry(nav_msgs/Odometry) from ZED stereo camera

- And you are going to combine these measurements to do sensor fusion by using Extended Kalman filter.

Requirement

- Try using package robot_pose_ekf to combine the measurements from IMU sensor and ZED stereo camera.
- Visualize and compare the path of ZED stereo camera, IMU integration and the result from package robot_pose_ekf in rviz.
- Trace the source code (**robot_pose_ekf.launch**, **odom_estimation_node.cpp**), and answer the questions.

Combine the measurements from IMU sensor and ZED stereo camera

- We'll provide you a slightly modified package **robot_pose_ekf**, please use our package to build your workspace.
- Before you start this task, please have a look at this documentation. In addition, you can find the original package **robot_pose_ekf** in the website (Click github link and change to Branch: kinetic-devel).
- In 1.2, it provides a method to build the package. However, you don't need to follow the steps. You can put the package in [~/your_worksapce/src/] and build it by typing **catkin_make**.
- To install the library that package need:
  $ **sudo apt-get install ros-kinetic-navigation**

Combine the measurements from IMU sensor and ZED stereo camera

- In robot_pose_ekf documentation, the node would subscribe some messages. In our case, we provide **/imu/data** (sensor_msgs/Imu), and **/zed/odom** (nav_msgs/Odometry).

- You will find that the messages' name which it subscribes are different than ours, thus it need some modification.

Map visual odometry to vo

- First, please have a look at file **robot_pose_ekf.launch** in the package we provided.

- Compare to the original version, we made some modification to inform the node that message **visual odometry** and **vo** are the same.

Republish a new imu data

- For imu data, you need to finish the following steps:
  1. Subscribe ***/imu/data***
  2. Transform ***/imu/data*** from IMU's frame to ZED odometry's frame
  3. Publish a new IMU data (transformed) named ***/imu_data***

How to transform ***/imu/data*** from IMU's frame to ZED odometry's frame

- A transform between two frame can be divided into two part: Rotation and Translation.
- Since IMU only provide the direction of movement, we don't need to take Translation into consideration here.
- That is, you only need to rotate original ***/imu/data*** into ZED odometry's orientation.

How to transform */imu/data* from IMU's frame to ZED odometry's frame

- The rotation matrix from IMU's frame to Camera's frame:

$$\begin{pmatrix} 0.0225226 & 0.999745 & 0.0017194 \\ 0.0648765 & -0.00317777 & 0.997888 \\ 0.997639 & -0.0223635 & -0.0649315 \end{pmatrix}$$

- The rotation matrix from Camera's frame to ZED odometry's frame:

$$\begin{pmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{pmatrix}$$

How to transform */imu/data* from IMU's frame to ZED odometry's frame

- Therefore, please use the rotation matrix to rotate IMU's data (orientation, angular velocity, linear acceleration velocity) into ZED's frame.
  (**Hint**: For orientation, you can change quaternion representation into matrix representation first. Both *Eigen::Quaterniond* and *tf::quaternion* support this transformation)

Combine the measurements from IMU sensor and ZED stereo camera

- After your node is able to publish ***/imu_data,*** type following command to launch the node:
  $**roslaunch robot_pose_ekf robot_pose_ekf.launch**
- Since it takes time to estimate new pose, please set rosbag play rate as 0.1 when you are running this node.
  $**rosbag play sdc_hw3.bag -r 0.1**
- After you launch the node, please subscribe this topic **/robot_pose_ekf/odom_combined** and draw the path according to the pose. (No need to consider covariance.)

Visualize and compare the path from different topic

- In this task, you need to visualize the path from ZED stereo camera, IMU integration and **robot_pose_ekf** in rviz.

- For ZED stereo camera, ***/zed/odom*** is of data type nav_msgs/Odometry. Please subscribe topic ***/zed/odom*** and use the position provided to draw a path. (No need to consider twist or covariance.)

- For IMU integration, you can transfer the result of assignment 3 into ZED odometry's frame and draw it.

- For **robot_pose_ekf**, after you launch the node, please subscribe this topic ***/robot_pose_ekf/odom_combined*** and draw the path according to the pose. (No need to consider covariance.)

Questions

Q1. What's the difference between our launch (**robot_pose_ekf.launch**) file and original launch file? And please explain why we add these modification.

Q2. Which parts in IMU data and ZED odometry are used? And please explain why it choose this way.(**odom_estimation_node.cpp**)

Q3. Please try to adjust covariance setting in odom_estimation_node.cpp (in *imuCallback()* & *voCallback()*), and observe how it affect the resulting path. Also, give your opinion which setting Is better, and why?

Q4. Comparing the resulting path and the single sensor paths, what is the difference, and why?

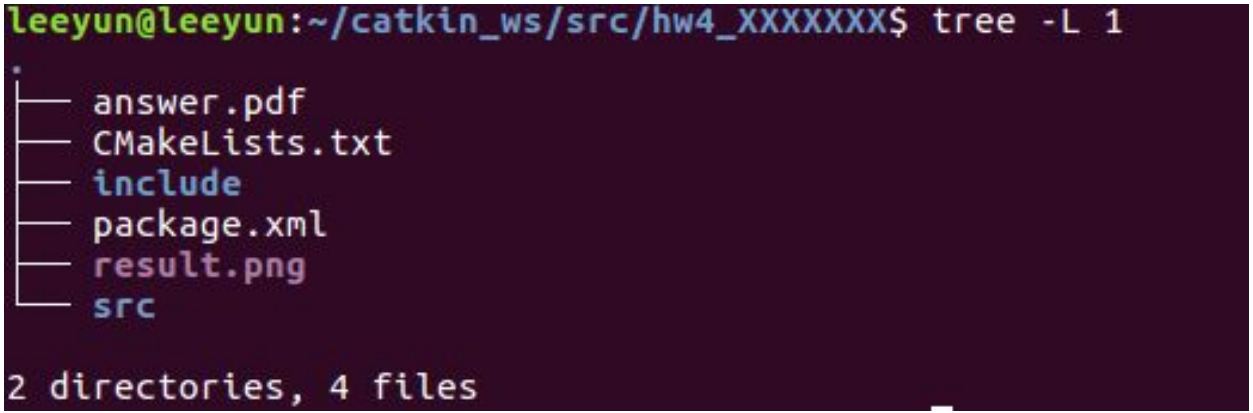
Submission Format

- Your program should publish three visualization_msgs/marker, they are the path of IMU integration (blue), visual odometry (red) and combined odometry (green) respectively.
- You can name the marker topics whatever.
- Name the package hw4_<student_id> and the executable hw4_node.

Submission Format

- Compress your file to hw4_<student_id>.zip(or tar, rar…etc.)
- In the zip file, it should contain:

1. Entire package with formulated name
2. The screenshot of the path visualized on rviz
3. Question answer in pdf format.



Submission Format

- Your screenshot may look like this:
  visual odometry(red)
  combined odometry(green)
  IMU integration(blue)
- Grid size = 1 m